SOA Integration Modeling: An Evaluation of How SoaML Completes UML Modeling

Todoran, Irina ; Hussain, Zuheb ; Gromov, Niina

Abstract: With the new trend of shifting from traditional architectures towards Service-Oriented Architectures (SOAs) today, the need to model integration becomes increasingly apparent. This study analyzes two main approaches for SOA integration modeling: using Unified Modeling Language (UML) and Service-oriented architecture Modeling Language (SoaML); having as a fundament a literature study, an evaluation between the two is made, based on a defined set of criteria. The results show where SoaML brings added advantages to UML and why it may be worth being used on a large scale.

DOI: https://doi.org/10.1109/EDOCW.2011.48

2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops

SOA Integration Modeling: An Evaluation of how SoaML Completes UML Modeling

Irina Todoran, Zuheb Hussain and Niina Gromov
Department of Computer Science and Engineering
Aalto University – School of Science and Technology
Espoo, Finland
{itodoran, zhussain, ngromov}@cc.hut.fi
*The first two authors contributed equally.

***Abstract*—With the new trend of shifting from traditional architectures towards Service-Oriented Architectures (SOAs) today, the need to model integration becomes increasingly apparent. This study analyzes two main approaches for SOA integration modeling: using Unified Modeling Language (UML) and Service-oriented architecture Modeling Language (SoaML); having as a fundament a literature study, an evaluation between the two is made, based on a defined set of criteria. The results show where SoaML brings added advantages to UML and why it may be worth being used on a large scale.**

Keywords: SoaML modeling, UML modeling, SOA integration modeling, modeling languages, service-oriented architectures modeling, service-oriented architectures integration.

I. INTRODUCTION

One of the major concerns of any organization executives nowadays is to integrate information systems to support the business strategies. An appropriate way to achieve this goal is to apply service-oriented architectures, which provide a flexible type of design principles for the integration of enterprise applications. SOA makes this possible by allowing for agile, scalable and controlled Enterprise Application Integration (EAI) solutions and providing a rather loosely-integrated suite of services [1,2,3]. Therefore, more and more organizations have started to migrate their architectures from classical to service-oriented. However, the implementation of SOA is much more challenging because of the complexity of the business functionality specific to enterprises. With this new approach, each business process needs to have a corresponding appropriate allocated service which is able to communicate with the others in a loosely coupled manner [2]. Breaking down the business into well-defined business processes which can be mapped to independently deployable services requires a comprehensive method of modeling, which these days is also increasingly gaining interest among researchers [3].

UML has been applied widely as a general purpose modeling language to model SOA, and create visual models with the set of graphic notations provided [4,5]. At the beginning, this seemed to be the most feasible solution for SOA integration modeling. Nevertheless, the high degree of generality of the method and the fact that it was initially thought as a language to work with artifacts of an object-oriented software-intensive system have conducted to the question whether there is a need for a dedicated, specific language for SOA modeling. This is how SoaML emerged: as a UML profile and metamodel for the modeling and design of services within a service-oriented architecture [4,5].

A. Research Question and Objective

Our main research question is "What benefits does SoaML bring compared to UML in the context of SOA integration modeling?" It will be answered in this paper by achieving the main objective: after having explained UML and SoaML in the context of service-oriented architectures, an evaluation of SoaML's capabilities against UML's capabilities to model SOA is made and conclusions are drawn regarding the importance of the differences between the two modeling languages.

B. Scope of the Research

The papers studied were chosen on the basis of their relevance to the evaluation carried out between the two approaches: using the existing UML notations for SOA modeling or using a new dedicated modeling language for SOA – SoaML. Moreover, the SoaML standard is rather new and largely neither implemented in practice nor researched by academia. Thus, the scope of our research is restricted by the materials which were to be found at the moment the present research was conducted.

C. Research Method – Literature Study

In order to fulfill the objectives of the research, the method which was chosen was literature study [6]. This assisted in collecting and structuring the information which exists in the field of SOA integration modeling and then in evaluating the two main approaches studied: modeling SOA with UML and SoaML.

The main references for this research paper were collected from ACM (www.acm.org), IEEE (www.ieee.org), OMG (www.omgwiki.org/SoaML) and IBM Corporation (www.ibm.com/developerworks/). They range from journal articles and conference papers to technical specifications provided by OMG (Object Management Group).

The technical specifications documents were chosen in order to ensure an accurate representation, according to the standards, and were studied for both UML and SoaML. The journal and conference papers were selected based on keywords search on the engines on the portals of the

978-0-7695-4426-7/11 $26.00 © 2011 IEEE
DOI 10.1109/EDOCW.2011.48

publishers and manual search in Conference Proceedings. The keywords used included: “UML modeling SOA”, “SoaML modeling SOA”, “SOA modeling”, “service-oriented architectures modeling”, “UML service specification”, “SoaML service specification”, “SoaML”.

The technical specifications were prioritized over the other types of papers when used as references for the presentation of the general features of the modeling languages, whereas the journal and conference papers were considered more relevant for supporting the examples where these two languages are used in real practice.

D. *Structure of the Study*

The first section introduces the background of the topic, research question, objective of the research and the research method used for the present study. The second section discusses the general aspects of integration modeling for service-oriented architectures, why modeling is necessary in this context, what the possible approaches are and why we are discussing the two (UML and SoaML) in this study. Then, Section 3 introduces UML as a modeling language for SOA integration and discusses to what extent and how this meets the specific needs of service-oriented architectures. SoaML as a modeling language for SOA integration is presented in Section 4, and the fifth section is dedicated to describing the benefits and limitations of the two languages studied: UML and SoaML. In the Discussion section, the results of the study are structured in the form of a table and commented upon. Thus, following the defined criteria, SoaML is evaluated against the UML. Section 7 draws the conclusions of the paper and the last section discusses some possible limitations of the research and potential directions for the future.

II. SOA Integration Modeling – General Aspects

In this section, the service-oriented architecture is introduced. It is also discussed how SOA differs from the traditional software architecture and how it is structured. Then, modeling of systems integration is described and the work towards modeling SOA is introduced.

A. *General Overview on SOA*

SOA is a framework enabling automation of services in a software system. The services are provided either to end user applications or other services distributed over a network, using published and discoverable interfaces. SOA provides flexible infrastructure and processing environment. This is done by providing independent, reusable and automated business processes, so that it is faster and easier to assemble new business processes and services for new business needs [7].

There are three distinctive characteristics that differentiate SOA from traditional software architecture:

1. The service elements must be ccoarse grained, loosely coupled and Autonomous.
2. The interaction between services is standards-based.
3. The services can be composed, found and replaced fast (that is, faster than in the traditional architectures), so that the structure and behavior of the architecture can be changing constantly [8].

The distinctive characteristics described above depict the evolution of software architecture towards SOA. SOA is predominantly about distributed applications and has three main parts: a provider, a consumer and a registry [9]. This simple architectural view of SOA, the web services architectural model, is presented in Figure 1.

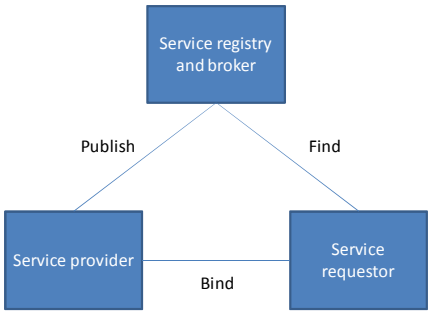

Figure 1. Web services architectural model [9].

The key element in Figure 1 is the service registry and broker. It is the facility where the service provider publishes its services and where the service requestor can discover services, finds all the information needed for the use of the service and binds to it [9].

B. *Integration Modeling*

Especially within enterprises, much interaction is required between systems to interact and fulfill complex business needs. Organizations opt for different techniques by which systems integration can be possible. Due to the diverse nature of businesses, there are numerous heterogeneous systems which exist in big enterprises today – some of them are legacy whereas others developed from scratch. Integrating such systems and identifying the potential interactions between them, when there is often insufficient documentation available, is a difficult task and requires careful consideration over the different important aspects of modeling such as: collaboration, semantics, syntax, role binding, behavioral modeling, level of abstraction. The discussion section of this paper utilizes these essential aspects to compare UML with SoaML.

Moreover, modeling is a critical part of information systems integration. Models allow for describing and abstracting the implementation of the system and the responsibilities for different parts of it. Model driven engineering (MDE) is a general software development methodology often also referred to as Model Driven Development (MDD) and Model Driven Software Engineering (MDSE). It is an approach where the use of models is the basis of the software development process. Whereas models have always been used in documenting software development, the MDE approach moves further by requiring the construction tools to be directed by models, too [10].

The best known MDE initiative is the Model-Driven Architecture (MDA) by OMG. MDA is a software design approach that was built around MDE principles and existing modeling standards, such as UML, XML Metadata Interchange (XMI), Meta Object Facility (MOF), etc. On a general level, MDA covers models for analysis and design, requirements and code [10].

There are many different standards and languages supporting the MDA. Probably one of the most mature (the standard is well known and there are numerous tools available) and commonly used modeling language is UML. UML was developed for general modeling of object oriented software system development. Because of the special nature of SOA, there were some issues in trying to model SOA integration with UML.

Because the aim of SOA is to dynamically fulfill the business needs, IBM suggested that the modeling of the architecture should be done from business models to SOA by the use of service modeling. Therefore, IBM developed the Service-Oriented Modeling and Architecture (SOMA) to complement UML. Since many businesses already model their business processes using UML, it is a good starting point for SOA modeling, too. When SOMA was developed, it was considered that combining elements from different methodologies and techniques is a good enough approach for modeling services [11].

However, UML eventually proved to have some limitations regarding SOA modeling. For example, the model management is done with the use of packages, which manage model elements by ownership, that is if an element is placed into one package, it cannot exist in another package (even though the other package might be able to access it, it does not always visualize the situation well enough). This problem was later solved in UML 2.x versions by the use of parts and connectors [12].

Since UML 2.x still was not enough for SOA modeling, it was extended by OMG and SoaML was born. SoaML supports the SOA approach to architecture more, addressing the concerns on what needs to be done, how it is done, where it gets done and who or what does it. In addition, technology-focused views of distributed computing are supported by SoaML [4,13].

Naturally, SoaML is not the only approach taken towards modeling SOA; there have also been smaller initiatives and projects attempting the same, but the results have not been as widely adopted as SoaML. Therefore, we chose not to look deeper into the other approaches in this study. The interest in SoaML is understandable, considering how widely spread and well known UML nowadays is. As UML has already been used in attempting to model SOA, it is interesting to find out how many benefits the new SoaML brings. In the following two sections, both UML and SoaML are described in more detail and how these are used for SOA integration modeling.

III. UML FOR SOA INTEGRATION MODELING

In this section, we introduce UML as a modeling language for SOA integration and to what extent and how this meets the specific needs of service-oriented architectures.

A. UML – General Overview

Modeling, by the means of diagrams, has always been highly used in various fields, ranging from construction, general engineering to software systems designs and even art. This has happened due to the three key benefits of modeling: clear communication, easy visualization and complexity management.

UML is a visual standardized general-purpose modeling language in the field of software engineering [4,5], used for specifying, constructing, and documenting the artifacts of systems. One can use UML with all processes, throughout the development lifecycle, and across different implementation technologies. UML was created by OMG as a standard in 1997. Over the past few years there have been minor modifications made to the language. UML 2 is the first major revision to the language and includes a set of graphic notation techniques to create visual models of software-intensive systems. The second version of UML is a major evolution in visual modeling. The new enhancements allow this new version to describe many of the elements found in today's software technology as well as the MDA and SOA [5].

B. Role of UML for SOA

Having as pre-requisites the need to model integration and the availability of a standardized way (UML), the modeling issue was soon seen as solved by applying UML for service-oriented architectures. For this, there are several approaches which are introduced in this section.

The 4+1 Architectural View Model

To begin with, SOA is a concept which refers to architectures in the first place, and therefore we find it appropriate to start our study by relating this to the generally accepted terminology and practices used in the field of software architectures. In this respect, the 4+1 Model was chosen as a reference [14] in order to show how the different views can be mapped to UML diagrams. Chung et. al's work [15] in this field is particularly relevant since they managed to show how each view can have its own associated UML diagrams and what they are used for. This is shown in Table I.

The Scenarios (or Central) View is used to describe architectural elements, validate architecture design and thus often assists when implementing prototypes. Among the UML diagrams, the most popular ones for this view are the Use case diagrams. The Logical View mostly deals with functionalities and therefore Class diagrams, Sequence diagrams and Communication diagrams are often used.

TABLE I. THE RELATIONSHIP BETWEEN 4+1 VIEWS AND UML DIAGRAMS (ADAPTED AFTER [15])

View	UML Diagram
Scenarios (Central)	- A use case diagram for use cases and their actors, such as a service consumer, service brokers, and service producers
Logical	- A class diagram for web services and their methods - A class diagram for XML message scheme
Development	- A sequence diagram for message exchanges between web services - An activity diagram for workflow
Process	- A component diagram for composite services and their partner web services with their URLs
Physical	- A deployment diagram for nodes and their networks of ESB with service components

The Development View, also known as Implementation view, usually serves the developers' (programmers') modeling needs. It uses the Package and Component diagrams. The Process View depicts the interactivity within the system and the runtime behavior. Therefore, Activity diagrams are used to a large extent. The Physical View uses the system engineer's perspective, dealing with the communication issues. The most common diagrams are the Deployment diagrams [14,15].

UML and Service Components

One modeling approach derived from this architectural kind of thinking is the one suggested by Stojanovic et. al [16] who consider using a domain and use cases in order to build UML diagrams, which are to be further used in the modeling process. In this respect, they put emphasis on two distinct primordial elements: UML language and the concept of service component. Thus, they define the service component as consisting of three main parts: context, contract and content (realization) [16]. Therefore, it is apparent that they realized the impossibility to only model SOA with UML and the obvious need to add something specific to the service-oriented architectures, which is the service component in this case. Moreover, the Coordination Manager service component is regarded as an essential element of each Business Service Component (BSC), with the role of managing and controlling how the Application Service Components (ASCs) interact to produce a higher level business goal specified by the contract of the BSC that encapsulates them. Nevertheless, they conclude that although there are established practices that cover certain aspects of the modeling and architectural design of component based solutions, the SOA modeling still poses certain requirements on top of it. Thus, directly applying UML concepts for modeling SOA, although it can be regarded as a good starting point, is not an entirely feasible approach [16]. In order to solve this issue, they propose a paradigm shift from components as objects to components as service managers, but this is yet to be put in practice.

Similarly, the service component is also seen as a "first-class modeling entity" when a modeling framework for service-oriented architectures is proposed. Giving the example of a supply chain management system, Zhang et. al claim the importance of implementing SOA in order to decrease the dependency between different software artifacts by promoting a loose-coupled and coarse-grained architecture [3]. And, in order to do this, services involved are first elicited from the business requirements and then the UML modeling comes in place.

More than UML: UML Profiles

When UML was published, it was already known that although it was wanted to be a standard, it would not perfectly match the needs of every organization and every project. Consequently, OMG published also an extension mechanism for the notations so that any organization can easily evolve in an agile and responsive manner that would allow the definition of new notations that will serve a new domain of interest. This extension mechanism is called UML Profiles [19]. Both researchers and industry practitioners have put much effort into creating new profiles for their own requirements, depending on the specificity of the work carried out.

For instance, Wada et. al [20] proposed a new UML profile for modeling non-functional aspects in service-oriented architectures. This graphically specifies and maintains non-functional aspects in SOA in an implementation-independent manner, and builds upon the existing UML elements precisely describing domain-specific and application-specific concepts simultaneously.

In addition, Lopez-Sanz et. al [21] proposed a Platform Independent Models (PIM) – Level UML profile for SOA, which includes specifying all the stereotypes for the new UML profile, which can be further used in this very specific context, being tailored for those PIM-Level concrete needs.

Another example of newly defined UML profiles is provided by Amir and Zeid [19] in their work, where they suggest using five distinct profiles: the resource profile, the service profile, the message profile, the service policy profile and the agent profile.

The focus of the resource profile is on using one specific key of the resource (URI, for instance) for modeling. This is extended by the service profile, by adding a Controller which assists in packaging different elements into a service component. Moreover, discovery service is also seen as a service component. The message profile is used for message exchange and deals with actual addresses (HTTP os SMTP, for example), while the service policy profile governs the way policies are modeled. The aim of the agent profile is to describe how the agents dealing with the service should be introduced into the model [19].

UML-RT (UML Real Time)

Having realized that UML alone is not enough for modeling architectures, researchers and industry practitioners [22-25] tried to use diverse UML variants of

extensions. In this respect, those worth mentioning are UML-RT and UML-S.

One problem with UML is that sequence diagrams are anonymous and therefore they do not allow referencing in other parts of the specification. Moreover, their syntactic means for expressing alternatives and repetition are limited, and notions such as hierarchy or components are absent. Therefore, binding of roles to concrete objects is only possible by sub-classing which, needless to say, is not a convenient approach for larger systems where there are objects with several roles. One way to address this issue is to use UML-RT, which brings the necessary hierarchical component model [22,23]. Nevertheless, the main problem related to this is the lack of continuity provided by the UML-RT framework from specifying services to their actual implementation.

UML-S

UML-S extends UML 2.0 activity diagrams to support 11 flow control patterns instead of the five basic ones [17] mentioned above [25]. They also allow to model calls to other Web services as well as the data transformation which may be required between the invocations. In addition, the UML-S framework is able to generate low-level code such as BPEL from high-level UML-S models.

IV. SOAML FOR SOA INTEGRATION MODELING

The main objective of this section is to give a general overview of the SoaML and to describe its role in modeling SOA. The section starts with the background and it also describes the new modeling capabilities and goals behind the SoaML.

A. SoaML – General Overview

SoaML is a recent OMG specification (first beta version released in April 2009), specifically designed to meet the requirements of SOA, and consists of a UML profile and metamodel for the specification and design of the services in SOA. SoaML does not make any changes to existing UML 2.0 notations, model elements, or semantics but adds the new notation, model elements and semantics to support the following new modeling capabilities [4,5,13]:

- *Identifying services* and the requirements they are intended to fulfill, and the anticipated dependencies between them.
- *Specifying services* through their functional capabilities. Specifying capabilities of the services which consumers provide, protocols to use them and the information exchanged between consumers and providers.
- *Defining service consumers and providers* through services they consume and provide, how they are connected and how consumers use the service functional capabilities and providers implement them in a manner consistent with specification protocols.
- *Defining policies* for using and providing services.
- *Defining service and service usage requirements*, ability to define services and service usage requirements and linking them to related OMG metamodels such as the BMM and BPMN.

The concept of SoaML revolves around the idea of services. It defines a service as: "A service is value delivered to another through a well-defined interface and available to a community (which may be the general public). A service results in work provided to one by another" [26].

Services architecture, service contract, service interface, simple interface and message type are the five key concepts of SoaML. *Services architecture* is a specification of a community, which defines its participants and their roles. It also defines the service contracts in the context of who provides and who consumes the services by defining collaboration. *Service contract* is the specification of the service which defines the roles of the service provider and its consumer; it also defines interfaces and choreography of services. *Service interface* defines bi-directional and *simple interface* defines one-directional service, and *message type* deals with the data which is exchanged between services [26]. SoaML, being a standard from OMG, complements the MDA techniques and provides an advantage where logical implementation of services can be treated separately from their physical realizations on different platforms [13]. Some goals for SoaML defined in its specification are [27]: provide complete support to model services in UML, support for bi-directional asynchronous services, support to model Services Architecture with multiple service providers and service consumers, support for services containing other services, compatibility with UML, BPDM and BPMN for business processes, support of direct mapping to web services, support for different modeling approaches (top-down, bottom up or meet-in-the-middle), ability to specify and relate the service capability and its contract.

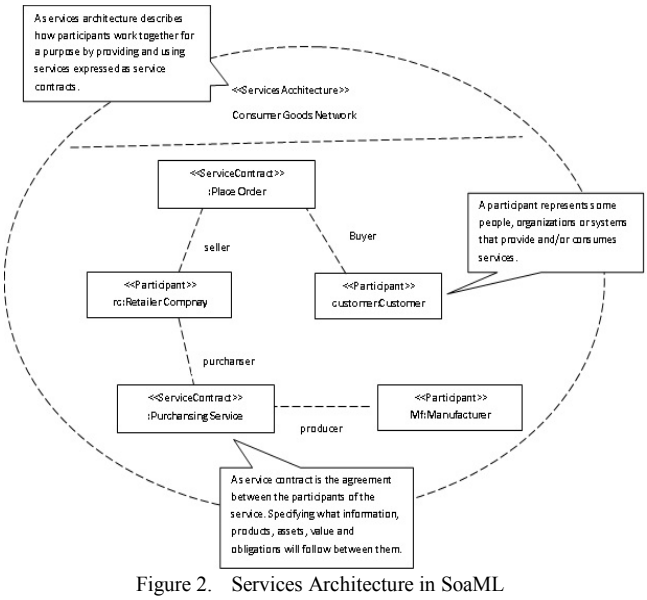

Figure 2. Services Architecture in SoaML

While UML comes with different profiles and modeling methodologies to support SOA modeling, SoaML provides a standard and specific way of modeling. Figure 2 shows a higher level diagram of the Services Architecture which presents how SoaML overcomes the limitations of simple UML by adding the new notations and semantics. The figure explains the Services Architecture for consumer goods network in which participants with different roles provide and consume services to fulfill the specific purpose. Participants in the Services Architecture are marked with <<Participant>> and Service Contracts with <<ServiceContract>>. Participants are connected to service contracts by a line which states the role of the participant in carrying out a service.

B. Role of SoaML for SOA

SOA is technology-independent and can be used to define not only information technology systems architectures, but also business architectures, mission architectures or community architectures, all based on the notion of service orientation. Designing service-oriented architectures is approach-independent, in a sense that there is no single way to build them [4]. The concept of SOA can be applied at the enterprise level where services are an abstraction of business processes or at the system level to understand and specify a particular service operation offered between systems [28,29].

According to OMG [4], SOA is an approach to systems architecture, where "architecture" defines how services and related parties can best work together to achieve specific goals and objectives. Here, architecture is more about a system of services which serve a community or organization rather than a single service. SoaML support for SOA defines how this system of services can be created for enterprises [4].

Today, there are numerous technology frameworks such as: .NET Framework, JSON-RPC-Java, AlchemySOAP or SOAP Lite which can be utilized to create web services. However, these frameworks are not suitable when high level concerns need to be described because of their limited ability to show the big picture of the interacting services within an enterprise. These frameworks are also limited in their functionality to model and support the SOA concepts of business services which are reusable, scalable, loosely-coupled, interoperable and composable [28,29].

SoaML is not limited to define only technology architectures although it fully supports the technology. SoaML services can be used to produce the code for the applications running in the enterprise. Services developed in SoaML can be used to generate artifacts such as: XSD, WSDL, Java, Ruby, C#, BPEL, HTML and Deployment Descriptors. Generated artifacts can be further used by developers as templates for service implementation or some artifacts can sometimes be complete enough to be executed directly. One of the strengths of SoaML is its scope. SoaML empowers the architects to create the complete service-oriented architecture for any enterprise: an architecture based on services where people, organizations and systems collaborate using those services, and services connect to other parts of the architecture, such as processes, information and business rules [28,29].

V. Benefits and Limitations of UML and SoaML

In this section, the benefits and limitations of UML and SoaML are presented, as they result from the literature study. Then, these will be assembled together in the form of a comparison table in Section VI.

A. Benefits and Limitations of UML

Since its appearance on the market, UML has always been of interest and increased in popularity due to the fact that it is more than a notation method, but a real language. It encompasses both semantics and syntax and can be used for modeling almost any type of system. Therefore, it has been able to serve real needs and it has never been only a theoretical concept.

In general, the main advantages UML are related to its ease of use and syntax that can apply to numerous situations, irrespective of we need to model an application (running on any type and combination of hardware, operating system, programming language, and network) or even middleware and systems. In addition, UML Profiles (the subsets of UML tailored for specific purposes) enable modeling Transactional, Real-time, and Fault-Tolerant systems naturally [4,5], making UML a language which is effective for modeling large, complex (software) systems. In addition, although getting familiar with UML is not a very complicated task, the features provided are very appropriate for specifying systems in an implementation-independent fashion, which is undoubtedly a plus.

Starting from all these benefits, industry practitioners and researchers commonly suggested that it is also an appropriate language to be used for modeling service-oriented architectures integration. In this respect, interaction diagrams can be used, profiles are available for customization according to the specific needs and, needless to say, it is a language generally accepted as the de-facto standard for modeling, which makes understanding and accepting it easier.

Although there are several generally accepted benefits of using UML for service-oriented, the fact that it was initially built upon the MOF™ metamodel for object oriented modeling constitutes a limitation when we discuss about it as a language for modeling architectures, and not necessarily software systems. In this respect, there are syntactic elements lacking and the linguistic incoherence is a term often used to express this as a disadvantage [3]. In addition to these, capabilities of UML and implementation language mismatch together with dysfunctional interchange format [24] are other limitations in this context.

What is more, as previously mentioned, service-oriented architectures are not an entirely technical concept, but

business plays a significant role in the way services are modeled. In other words, there will most often be a service designated to address each business process and this is where UML does not seem to be enough. It does not provide anything specially designed for business rule specification. Although a group exists for this within OMG, they have not published anything official so far and this may only be included as part of a future extension of the language.

Besides these, there is evidence that it is rather poor for distributed systems – there is no way to formally specify serialization and object persistence [30], and this is obviously a need for modeling service-oriented architectures. For instance, it is not possible to specify that an object resides on a server process and is shared among instances of a running process with the current definition of UML language.

B. Benefits and Limitations of SoaML

The concept of SOA is not new and has been promoted for several years. Different approaches have been taken to model service-oriented architecture including UML, but most lack essential SOA attributes. In 2009, OMG proposed SoaML, a dedicated language specially designed for service-oriented architecture modeling. SoaML provides special support to cater service modeling and design activities, and it takes the model driven development approach for SOA modeling. The specification of systems of services, the specification of individual service interfaces, and the specification of service implementation are few of the modeling requirements of service-oriented architectures which are currently supported by SoaML.

According to the SoaML specification by OMG, SoaML is a language to support service-oriented architectures. It does not specify a particular development methodology, which is the reason why different methodologies are being proposed nowadays [31]. Nevertheless, SoaML itself offers various benefits and some are the following [32-37]: it enables the service interoperability and integration at the model level, provides support to model services at a higher level of abstraction without being concerned about the lower level technological details, separates the business integration and service interaction concerns at the architectural level, enables the SOA both on and between existing platforms through the MDA, decouples the solution architecture platform implementations to prevent existing solutions from inhibiting platform evolution, leverages and integrates with existing OMG standards .

In the context of SOA modeling for enterprises, SoaML provides a way to model service-oriented architectures in a standardized and platform-independent way. After service-oriented architectures are modeled with SoaML, those architectures can be implemented directly by using automated tools. This provides the enterprises with agility, collaboration and efficiency when developing SOA. The separation of business and technology concerns by identifying services through business processes and using model driven approaches helps to better understand and meet the needs of enterprises, as it provides and uses services in the supply chain, between different departments, units and divisions. With the extensive support for SOA modeling, SoaML provides a way to model Service Contracts and Service Interfaces, to define the roles, interfaces and message data of services within the enterprise. It also provides the flexibility to model services of various complexities and with different scopes. With SoaML, enterprises can model services from a simple service operation to rich bi-directional and asynchronous enterprise scale services. At the enterprise level, the capability of SoaML to model Services Architectures helps to show how the systems of participants and components provide and use services to achieve business value. Also, the automation of the development, testing and maintenance processes with MDA reduces the development and maintenance costs while improving the agility of the enterprise. Moreover, when SOA is modeled with SoaML, the clear understanding of organizational processes and structure can help to improve SOA governance within the enterprise [28,29].

Apart from the benefits which SoaML brings, there are still some areas which are not supported. Providing the full support for modeling SOA, it still lacks the support for styles or configurations. The scope of the SoaML specification does not directly cover the SOA governance and compliance issues, service quality, reliability of message delivery, wire-level protocols, service brokering, and others. Since SoaML is still at an incipient age, it can be expected that as it grows it incorporates these missing areas. But there is also a possibility that SoaML will be integrated in other standards which already cover these aspects [4,5].

There are also other features of SoaML which are highly criticized. According to Poulin [38], SoaML is not about orientation on service and is not a complete architecture modeling language because it does not model the architecture entities in full, but concentrates on the relationships between them (which is important, but not enough). The definition of SOA given by the SoaML specification puts emphasis on "defining how people, organizations and systems provide...services", which is a really new “aspect of architecture”. The stress that SoaML puts on Participant can be seen as very confusing. It appears that the relationship between consumer and service provider is more important than the relationship between consumer and service itself in SoaML [38].

VI. DISCUSSION

The aim of this section is to assemble together the findings of the present study, answering our research questions, with emphasis on the evaluation of how SoaML completes UML modeling and the most significant positive aspects of SoaML. In order to accomplish this, eleven criteria were formulated to evaluate the modeling aspects of the two languages: semantics, behavioral diagrams for collaboration, syntax, referencing mechanism, service

concept, binding of roles, component notion, SOA Governance support, level of abstraction, integration flexibility, decoupling of solutions and business architecture. The criteria were selected based on the relevance for SOA integration modeling, the frequency the issues were discussed in the papers studied and the importance allocated to these by the authors. Moreover, it is worth mentioning that the viewpoints belong to both academic researchers and industry practitioners who need to deal with SOA integration modeling.

TABLE II. COMPARISON BETWEEN SOAML AND UML

Language / Criteria	SoaML	UML
Semantics	Formal semantics existing [24]	No support [24]
Behavioral Diagrams for Collaboration	Any kind of behavior admitted [4,5,13]	Only interaction diagrams are regarded as suitable [4,5,24]
Syntax	Despite being a dedicated language for SOA, SoaML is based on the UML profile and metamodel [4,5,13]	Defined by meta-modeling, difficult to check [24]
Referencing Mechanism	Named diagrams [4,5]	Anonymous diagrams, impossible to refer elsewhere [22,23]
Service Concept	Extends UML and provides the concept of Service Participants, Services Interfaces, Service Contracts and Service data [4,5,13]	Non-existing as an independent, abstract modeling element [22,23]
Binding of Roles	Service specifications need not to be instantiated to concrete objects [4,5,13]	Roles can be bound to concrete objects by sub-classing, resulting in cluttered specifications [22,23]
Component Notion	Existing as a stand-alone concept, referring to components of the architecture and not necessarily to objects as in the object-oriented paradigm [4,5,13]	Component diagrams only refer to implementation components [22,23]
SOA Governance support	Understanding of organizational processes and structure can help to improve SOA governance [28,29]	No support for SOA Governance
Level of abstraction	Models the services at higher level of abstraction without technology and underlying platform concern [33-37]	No support for modeling services; the "service" concept is absent
Integration flexibility	Can be easily integrated with other OMG standards like MDA or BPMN [4]	No integration support with other OMG standards
Decoupling of solutions and business architectures	Allows flexible platform choices and prevents existing solutions from inhibiting platform evolution [33-37]	No support

The first criterion considered is **Semantics.** In their very recent work [24] on revising UML collaborations, Astesiano and Reggio conduct a thorough analysis on the problems with UML collaboration and semantics. This is mostly associated with the lack of consensus on the type of behavioral diagrams which can and need to be used for semantic collaborations. Whereas UML provides no support in this respect, SoaML comes with in-built mechanisms for formal semantics. However, it is important to note the fact that semantics is still regarded as a young field of research, which is not to be found implemented in practice on a very large scale.

Behavioral diagrams for collaboration. According to OMG [4,5], when a service-oriented architecture is modeled with UML, interaction diagrams are the only ones which are suitable for the modeling process of behavioral collaboration. This point is also emphasized by Astesiano and Reggio [24], and regarded as a serious limitation of UML modeling for SOA. In contrast, when SoaML is chosen as the modeling language, any kind of behavior is admitted, which leads to more flexibility and availability of more options. A new conceptual model for collaboration and new relationships are defined in SoaML [24], which makes the relationships much easier to model, compared to the UML old-style.

Syntax. SoaML is a dedicated language for service-oriented architectures whereas for UML, the syntax is defined by meta-modeling and considered difficult to check. However, the SoaML solution is also based on UML profile and meta-model [4,13], which makes this only a minor enhancement to UML.

Referencing mechanism. In UML, the sequence diagrams are anonymous, which does not allow referencing them in other parts of the specification. Moreover, their syntactic means for expressing alternatives and repetition are limited [22,23], in addition to the limitations of expressing the independent sending of messages, or composing a specification from parts. In contrast, SoaML brings a significant change by making it possible to have named diagrams, which can therefore be further referred in other parts of the model.

Service Concept. Service Concept is the basic motivation behind the emergence of SoaML as a new language. All the basics of service concept are well taken care of by SoaML, which makes it different from UML. It adds notations, model elements and semantics that extend the existing UML 2.x capabilities for service modeling. It introduces the concepts of Service Participants, Service Interfaces, Service Contracts and Service Data. On the other hand UML does not take into account the modeling aspects of services in SOA.

Binding of Roles. In UML, binding of roles is supported through the classes, whereas the binding of roles to concrete objects is only possible through sub-classing, which results in cluttered specifications [22,23]. In SoaML, role binding does not need service specifications to be instantiated to concrete objects – each participant represents a role [4,5,13].

Component Notion. In SoaML, the concept of component is not limited to the software component, but it exists as a stand-alone concept which refers to the components of architecture in SOA [4,5,13]. In contrast, UML has a narrow focus on the components notion – they are only considered software components from the implementation perspective and are modeled through class diagrams [22,23].

SOA governance support. Although not in a very direct way, SoaML provides governance support thanks to its MDA approach towards service modeling (by modeling services at business level and system level), and creates a picture of underlying organizational business processes and structures which can help to improve SOA governance [28,29]. UML, being very general in its modeling approach, lacks this capability.

Level of abstraction. SoaML methodology for service-oriented architectures shows how services can be modeled at different levels of abstraction (business, technology). Since services are generally designed after having identified businesses processes, it helps to model them at a higher level of abstraction without any technology or underlying platform concerns [33-37]. UML has no special support for modeling services at different levels of abstraction.

Integration flexibility. SoaML specification, being provided by OMG, has the advantage of being very compatible and well integrating with other OMG standards. It can be easily used with MDA or BPMN to support the enterprise SOA. UML does not have this flexibility as it lacks its support for SOA.

Decoupling of solutions and business architectures. SoaML, when used with MDA, provides the ability to model services by separating the concerns of solution and business architectures. This allows flexible platform choices and prevents existing solutions from inhibiting platform evolution [4,33-37]. UML does not provide this support.

VII. CONCLUSION

In this paper, we started by generally discussing integration modeling for service-oriented architectures and two possible approaches for this: modeling with UML and SoaML. After having separately presented how modeling takes place for both and what the benefits and limitations are, the evaluation is made between the two.

The main conclusions to be drawn are that although UML is very popular for software systems modeling and largely accepted as the standard for this, it still lacks numerous features that are necessary for service-oriented architectures modeling. Despite the fact that UML 2.0 brought numerous enhancements and the UML Profiles are there to assist in customizing UML for specific needs, it is apparent that these are not enough to model architectures. First of all, there are significant concepts missing – UML does not define the service or architecture component at all and, needless to say, this is one element of utmost importance for an SOA.

By contrast, SoaML provides support for most of the specific issues of SOA integration modeling and the specific syntax needed for architectures, greater flexibility, level of abstraction and a dedicated standard language.

To sum up, this study shows that there are strong chances that SoaML will be adopted by industry as a modeling language for modeling the integration of service-oriented architectures because it provides the specific support and syntax for SOA which are missing from UML, as it was shown in the Discussion section.

VIII. LIMITATIONS AND FURTHER RESEARCH

Possible limitations of the literature study can result from the rather low number of existing research materials about SoaML and the increased popularity of modeling with UML. Thus, the papers studied for the two methods were not perfectly numerically balanced.

The study can be continued by taking a more practical approach by modeling several cases of SOA integration using both methods, and trying to identify the benefits and limitations resulting from the concrete practical modeling.

ACKNOWLEDGMENT

The authors would like to thank Ilkka Melleri (SoberIT, Aalto University – School of Science and Technology) for his continuous support and constructive feedback provided throughout this project and for writing this paper.

REFERENCES

[1] J. Bih, "Service Oriented Architecture (SOA) – A New Paradigm to Implement Dynamic E-business Solutions. Ubiquity," Vol. 7, Issue 30, 2006.

[2] M. P Papazoglou and W. van den Heuvel, "Service oriented architectures: approaches, technologies and research issues," The VLDB Journal 16, Springer-Verlag, 2007, pp. 389-415.

[3] T. Zhang, S. Ying, S. Cao and X. Jia, "A Modeling Framework for Service-Oriented Architecture," Proceedings of the Sixth International Conference on Quality Software (QSIC'06), 2006.

[4] Object Management Group (OMG), "Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS)," 2009.

[5] Object Management Group (OMG), "Specification for the UML Profile and Metamodel for Services (UPMS), Service oriented architecture Modeling Language (SoaML)," 2009, available at: http://www.omg.org/spec/SoaML/1.0/Beta1/PDF/, accessed: November 2010.

[6] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," MIS Quarterly, Vol. 26, 2002, pp. 13-23.

[7] M. P. Papazoglou, "What's in a Service?," Lecture Notes in Computer Science, Volume 4758, 2007, pp. 11-28.

[8] X. Jia, S. Ying, S. Cao and D. Xie, "A New Architecture Description Language for Service-Oriented Architecture," Grid and Cooperative Computing, GCC 2007, Sixth International Conference, 2007, pp. 96-103.

[9] M. N. Huhns and M. P. Singh, "Service-Oriented Computing: Key Concepts and Principles," Internet Computing, IEEE, Vol. 9, Issue 1, 2005, pp. 75-81.

[10] I. Borne, T. Gherbi, and T. Meslati, “MDE between Promises and Challenges,” 11th International Conference on Computer Modelling and Simulation (UKSim 2009), 2009.

[11] O. Zimmermann, N. Schlimm, G. Waller and M. Pestel, “Analysis and Design Techniques for Service-Oriented Development and Integration,” IBM Deutschland, 2005.

[12] S. Johnston, “Modeling service-oriented solutions,” IBM Developerworks, 2005.

[13] The Open Group, “Appendix: Navigating the SOA Open Standards Landscape Around Architecture White Paper : Description of Targeted SOA Open Standards Technical Products,” 2010, available at: http://www.opengroup.org/soa/source-book/stds/desc.htm, accessed: November 2010.

[14] P. Kruchten, “Architectural Blueprints – The 4+1 View Model for Software Architecture,” IEEE Software 12(6), November 1995, pp. 42-50.

[15] S. Chung, S. Davalos, C. Niiyama, D. Won, S.-H. Baeg and S. Park, “A UML Model-Driven Business Process Development Methodology for a Virtual Enterprise using SOA & ESB,” IEEE Asia-Pacific Services Computing Conference (IEEE APSCC), 2009, pp. 246-253.

[16] Z. Stojanovic, A. Dahanayake and H. Sol, “Modeling and Design of Service-Oriented Architectures,” IEEE International Conference on Systems, Man and Cybernetics, 2004, pp. 4147-4152.

[17] R. Grønmo and I. Solheim, “Towards Modeling Web Service Composition in UML,” 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure, Porto, Portugal, 2004.

[18] P. Wohed, E. Perjons, M. Duma and A. ter Hofstede, “Pattern based analysis of EAI languages - the case of the business modeling language,” Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS), Angers, France, April 2003, pp. 174-184,.

[19] R. Ami and A. Zeid, “A UML profile for service oriented architectures,” OOPSLA 2004 Companion, Vancouver, Canada, October 2004, pp. 192-193.

[20] H. Wada, J. Suzuki and K. Oba, “Modeling Non-Functional Aspects in Service-Oriented Architecture,” IEEE International Conference on Services Computing (SCC’06), 2006.

[21] M. Lopez-Sanz, C. J. Acuna, C. E. Cuesta and E. Marcos, “Modeling of Service-Oriented Architectures with UML,” Electronic Notes in Theoretical Computer Science, No. 194, 2008, pp. 23-37.

[22] I. H. Krüger, “Specifying Services with UML and UML-RT – Foundations, Challenges and Limitations,” Elsevier Science B. V., Electronic Notes in Theoretical Computer Science, Vol. 65, No. 7, 2002, pp. 1-12.

[23] I. H. Krüger, “Towards Precise Service Specification with UML and UML-RT,” 2003.

[24] E. Astesiano and G. Reggio, “Revising the UML Collaborations: A Well-Founded Approach,’ C. Choppy and O. Sokolsky (Eds.): Monterey Workshop 2008, LNCS 6028, 2010, pp.1-23.

[25] C. Dumez, J. Gaber and M. Wack, “Web Services Composition using UML-S: a case study,” GLOBECOM Workshops, IEEE, 2008, pp. 1-6

[26] A. Berre, D. Roman, B. Elvesaeter and C. Carrez, “Service oriented architecture Modeling Language (SoaML),” Tutorial at MOPAS2010, Athens, Greece, 2010, available at: http://www.iaria.org/conferences2010/filesMOPAS10/MOPAS_2010_Tutorial.pdf, accessed: November 2010.

[27] J. Berre, “OMG SoaML Service oriented architecture Modeling Language – UML Profile and Metamodel for Services,” 2009, available at: http://www.uio.no/studier/emner/matnat/ifi/INF5120/v09/undervisningsmateriale/F04-090209-SoaML.ppt, accessed: November 2010.

[28] C. Casanave, “Enterprise Service Oriented Architecture Using the OMG SoaML Standard,” 2009, available at: http://www.omg.org/news/whitepapers/EnterpriseSoaML.pdf, accessed: November 2010.

[29] C. Casanave, “Enterprise-SOA with SoaML,” Data Access Technologies Inc., Model Driven Solutions, 2009.

[30] S. Kou, M. A. Babar and A. Sangroya, “Modeling Security for Service Oriented Applications,” ECSA 2010, Copenhagen, Denmark, August 2010, pp. 294-301.

[31] S. Dustdar and F. Li, “Service Engineering: European Research Results,” Strauss GmbH, Morlenbach, Germany, available at: http://books.google.com/books?id=piuG4bzu9doC&pg=PA25&dq=%2Bsoaml&hl=en&ei=JqfJTKSdHsLoOdDJwcMB&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCoQ6AEwAA#v=onepage&q=%2Bsoaml&f=true, accessed: November 2010.

[32] J. Bezivin, R. M. Soley and A. Vallecillo, Proceedings of the First International Workshop on Model-Driven Interoperability, Oslo Norway, ACM Press, 2010.

[33] J. Amsden, “Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 1. Service Identification,” 2010, available at: http://public.dhe.ibm.com/software/dw/rational/pdf/modelingwithsoaml-1.pdf, accessed: November 2010.

[34] J. Amsden, “Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 2. Service Specification,” 2010, available at: http://public.dhe.ibm.com/software/dw/rational/pdf/modelingwithsoaml-2.pdf, accessed: November 2010.

[35] J. Amsden, “Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 3. Service Realization,” 2010, available at: http://public.dhe.ibm.com/software/dw/rational/pdf/modelingwithsoaml-3.pdf, accessed: November 2010.

[36] J. Amsden, “Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 4. Service Composition,” 2010, available at: http://public.dhe.ibm.com/software/dw/rational/pdf/modelingwithsoaml-4.pdf, accessed: November 2010.

[37] J. Amsden, “Modeling with soaML, the Service-OrientedArchitecture Modeling Language: Part 5. Service Implementation,” 2010, available http://public.dhe.ibm.com/software/dw/rational/pdf/modelingwithsoaml-5.pdf, accessed: November 2010.

[38] M. Poulin, “SoaML is about everything but SOA,” Part 1, 2009, available at: http://www.ebizq.net/blogs/service_oriented/2009/06/soaml_is_about_everything_but_soa_part_1.php, accessed: November 2010.